# Advanced Machine Learning Algorithms

Assessment 3 | Travel Airfare Data Product

**Experiment Report** [XGBoost]

Nathan Collins

Nathan Collins | Yasaman Mohammadi | Agustin Ferrari | Divgun Singh
12062131 | 24612626 | 24704114 | 24586556

| | |
|---|---|
| **Assessment 3:** | **Modelling and Deriving Insights** |
| **Type:** | Group Assessment |
| **Deliverables:** | Pre-processing Jupyter Notebook (1x)<br>Modelling Jupyter Notebooks (x4)<br>Final Report **4000 words** |
| **Points:** | 100 |
| **Due:** | Sunday, 10th November, 23:59 |

## Assessment Criteria

- *Comprehensibility, quality, reliability, robustness, and readability of code (25 points)*
- *Insightfulness and quality of results including assessment, risks, and recommended next steps from a business point of view (20 points)*
- *Appropriateness of communication style to audience in final report (summarization, quality of findings, recommendations, visualisations, readability, clarity) (20 points)*
- *Breadth of evidence of collaborative work (e.g., meeting minutes, details of contributions, etc.) (20 points)*
- *Relevance of documentation and instructions for deploying models and running applications, and robustness of data services (15 points)*

# EXPERIMENT REPORT **[ 3 ]**

## EXPERIMENT BACKGROUND

### a. Business Objective

The objective is to build an application that provides users in the USA with quick and accurate price estimates for local flights. Users can enter details about a trip, and the app will immediately predict fares.

This product will assist in travel planning and budgeting, making flight costs clearer for consumers. This service endeavours to empower users to make more informed decisions, potentially leading to cost savings on travel expenses, and providing an improved understanding of fare trends across different times and routes.

### b. Hypothesis

**Null** | *The predictive model developed for estimating travel airfare does not significantly differ from random guessing or a simple heuristic-based on average fares.*

**Alternate** | *The predictive model provide significantly more accurate estimates of local travel airfare than random guessing or a simple heuristic-based on average fares.*

The null hypothesis assumes no relationship exists between the dataset features, while the alternative hypothesis suggests a relationship. By comparing the model's performance against the null hypothesis, it determines whether the model is significantly better than random chance at predicting customer purchase behaviour.

Investigating these questions will assist with accomplishing the business goal by determining which existing customers who have purchased a second vehicle share certain features.

### c. Experiment Objective

Experiment 3. will consist of an **XGBoost Regression** model, examining the **Target** variable ("totalFare") against all other variables independently.

## EXPERIMENT DETAILS

### a. Data Preparation

#### Understanding the Dataset

The dataset comprises `13521345` observations across `27` distinct variables.

Observations entail a history of airport-sorted flights across multiple companies, with features associated with each flight. These flights are pre-sorted by US-airports:

| | |
|---|---|
| **SFO**: | San Francisco International Airport |
| **ATL**: | Hartsfield-Jackson Atlanta International Airport |
| **BOS**: | Boston Logan International Airport |
| **CLT**: | Charlotte Douglas International Airport |
| **DEN**: | Denver International Airport |
| **DFW**: | Dallas/Fort Worth International Airport |
| **DTW**: | Detroit Metropolitan Wayne County Airport |
| **EWR**: | Newark Liberty International Airport |
| **IAD**: | Washington Dulles International Airport |
| **JFK**: | John F. Kennedy International Airport |
| **LAX**: | Los Angeles International Airport |
| **LGA**: | LaGuardia Airport |
| **MIA**: | Miami International Airport |
| **OAK**: | Oakland International Airport |
| **ORD**: | O'Hare International Airport |
| **PHL**: | Philadelphia International Airport |

Each dataset includes the target outcome "**totalFare**" ,

alongside other noteworthy traits, such as

- **flightDate**,
- **startingAirport** & **destinationAirport,**
- **travelDuration** & **totalTravelDistance**
- and segment data, such as **segmentsAirlineName.**

`12` of the `27` features are prefixed by "**segmented.**"
Each column represented multiple flight data, separated by II.

## Data Cleaning

### Reading the complete Dataframe

The data frame was formed with a script that navigated through a hierarchical file structure (list of zipped airport folders), which individually identified and extracted CSV files from each subdirectory. The CSV files were subsequently read into a pandas DataFrame, followed by concatenation into one comprehensive DataFrame, ready for the next stages of data processing.

### **Checking for** missing **values**

The dataset was first checked for missing values, where nulls discovered were only apparent in the totalTravelDistance and segmentsEquipmentDescription features. These columns were omitted prior to modelling [figure 1]. It was later discovered that segmentsDistance possessed values such as 'None || None', the equivalent to further null values. This feature was likewise omitted prior to modelling.

### **Processing “**segment**” Columns**

As 12 of the 27 features contained segmented flight data separated by “**II**”, each needed to be assessed for modelling relevance and handled individually. The most applicable of these were segmentCabinCode, which were encoded into individual features and assigned a percentage based on their apparency in a specific row.

For example: **coach || coach || business**

| coach | premium coach | business | first |
|---|---|---|---|
| 66.6% | 0 | 33.3% | 0 |

### **Encoding** departureDate **&** arrivalDate **Variables**

Features critical to the model’s input departureDate & arrivalDate were encoded into DepartureDateDayOfWeek and DepartureDateMonth.

The initial segmentsDepartureTimeRaw column contained values with distinct time entries separated by '||'. Initially, the column was divided to acquire the first datetime as it represents the departure date. After that, the day of the week and month were computed for each timestamp. An example of such a value is: '2022-05-30T15:00:00.000-04:00||2022-05-30T21:40:00.000-04:00'.

### **Encoding** isRefundable **to numerical**

isRefundable was transformed into a numerical format by mapping Boolean values (True and False) to integers (1 and 0). This encoding process ensures that the feature can be effectively used as numerical input for the model.

### Dropped Features

Total omitted features:

lagId,
isBasicEconomy and isNonStop:
totalTravelDistance and travelDuration:
‘totalTravelDistance, ‘
‘travelDuration', ‘segmentsDepartureTimeEpochSeconds',
'segmentsArrivalTimeEpochSeconds', 'segmentsArrivalTimeRaw',
'segmentsArrivalAirportCode', 'segmentsDepartureAirportCode',
'segmentsAirlineName',
'segmentsAirlineCode', 'segmentsEquipmentDescription',
'segmentsDurationInSeconds',
'segmentsDistance'

## b. Feature Engineering

Six supplementary features were added to assist in improving machine learning accuracy through magnifying the underlying necessity of the data. This was also performed to help mediate the exclusion of specific features, in order to meet the business requirements for a minimal input by the user.

These included;

**days_in_advance**
Examines how many days exist between the date of the flight's search and the actual flight date, (entails the sum of the no. days between the flightdate and searchDate) - included, as airlines often adjust pricing depending on how early or late a booking takes place.

**search_date**
Following extraction of the day_of_the_week , month , and year from flightDate, values were then converted into an integer format.

**flight_date**
Following extraction of the day_of_the_week , month , and year from searchDate, values were then converted into an integer format.

**departure_time**
Extracted from segmentsDepartureTimeRaw. Originally formatted as ('2022-05-20T18:58:00.000-07:00||2022-05-21T00:5...'), the 24hr time was isolated and converted to an integer format.

**segmentsCabinCode**
Extracted from segmentedCabinCode. This feature acts as an intermediate prior to converting cabin differentiations into percentages, (see 2.2 "Processing segment Columns").

**numSegments**
The count of stops in each trip, is derived from segmentsCabinCode.

**isRefundable**
We encoded this feature by converting its boolean values (True and False) to integers (1 and 0), ensuring it could be effectively utilized as numerical input for the model.

**startingAirport**
One-hot encoding was applied to this feature.

**destinationAirport**
One-hot-encoding was performed on this feature as well.

## c. Modelling

### Selecting a performance metric

The Root Mean Squared Error (RMSE) was applied to evaluate the XGBoost model due to its sensitivity to large errors, thus ensuring that the model accurately predicts not just the average trend but also the extremes. Meanwhile, the Mean Squared Error (MSE) was chosen as it directly measures the average squared difference between the estimated values and the actual value, providing a clear quantification of the model's prediction error.

### Establishing the model:

All models across all experiments, will be constructed through a Python function. This helps establish consistency across all metrics within the dataset and throughout the transformation of the data prior to and during machine learning. The subsequent model selected for this stage of the experiment is intended to expand upon Experiment 1’s insights and tangent away from a simple model.

**XGBoosting** remains a widely integrated algorithm for regression and classification modelling due to its ability to handle data variety, its efficiency, its performance and its scalability. It was selected as the next classification model due to its gradient boosting functionality, integrated regularisation to prevent overfitting, hyperparameter tuning functionality and ability to provide insights into influential features with “feature importance”.

**Hyperparameter tuning**:

- **learning_rate**: As it regulates the step size at which the model learns. This is important as the learning rate determines how quickly the model may adjust its parameters based on the error gradient during training. Lower learning rates, for example, aid in helping the model achieve accuracy through smaller steps.

- **max_depth**: Specifies the maximum depth of each decision tree in the ensemble, this helps reduce or increase complexity.

- **n_estimators**: The number of boosting rounds (trees) to integrate within. Increasing rounds, for example, integrates complexity though at a cost of potentially overfitting.

- **min_child_weight**: Uses “small child” nodes to prevent overly deep trees (minimum sum of instance weights in each child). Higher values help improve generalisation.

- **gamma**: specifies the proportion of features to be randomly sampled for building each tree, helping in regularisation and potentially enhancing model performance by reducing overfitting.

## EXPERIMENT RESULTS

### a. Technical Performance

#### Evaluation of precision

#### Performance Metrics: RMSE and MSE

- High RMSE or MSE: Indicates poor model performance, as the predictions are significantly deviating from the actual values. This implies that the model is not accurate in predicting airfare flights, leading to unreliable forecasts.
- Low RMSE or MSE: Suggests good model performance, with predictions closely aligning with actual values. This is desirable for predicting airfare flights, as it ensures more accurate and trustworthy predictions.

## Technical Performance & Analysis

### Baseline Model

**MSE**: 41820
**RMSE**: 204

**Summary**: The high MSE and RMSE indicate a significant error in the baseline model's predictions, necessitating improvements for reliable airfare predictions.

### Experiment 1

Summary Table:

| Model | Test MSE | Test RMSE | Train MSE | Train RMSE |
|---|---|---|---|---|
| **1** | 16534.6 | 128.59 | 15035.53 | 122.62 |

**Analysis**: Improvement over baseline with lower errors. However, train and test errors are close, indicating no overfitting but still a high error margin.

**Next Step**: Experiment with different hyperparameters to further reduce error.

### Experiment 2

Summary Table:

| Model | Test MSE | Test RMSE | Train MSE | Train RMSE |
|---|---|---|---|---|
| **2** | 17106.16 | 130.79 | 16194.68 | 127.26 |

**Analysis**: Slight increase in error compared to Experiment 1, suggesting that the new hyperparameter combination was less effective.

**Next Step**: Alter hyperparameters significantly to explore model improvements.

### Experiment 3

Summary Table:

| Model | Test MSE | Test RMSE | Train MSE | Train RMSE |
|---|---|---|---|---|
| **3** | 14425.48 | 120.11 | 7136.15 | 84.48 |

**Analysis**: Significant improvement in both train and test errors. However, the larger gap between train and test errors indicates potential overfitting.

**Next Step**: Address potential overfitting while maintaining or improving accuracy.

### Experiment 4

Summary Table:

| Model | Test MSE | Test RMSE | Train MSE | Train RMSE |
|---|---|---|---|---|
| **4** | 14079.04 | 118.66 | 4379.78 | 66.18 |

**Analysis**: Further reduction in errors with a continued issue of overfitting, as seen from the gap between train and test errors.

**Next Step**: Fine-tune hyperparameters to balance overfitting while maintaining low error rates.

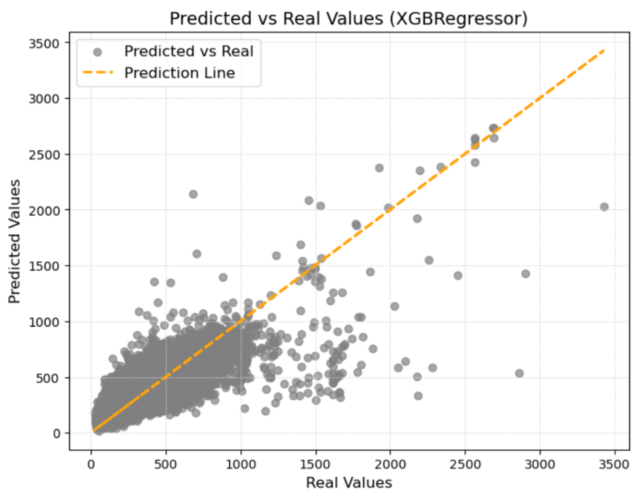
### Experiment 5 (Hyperopt)

Summary Table:

| Model | Test MSE | Test RMSE | Train MSE | Train RMSE |
|---|---|---|---|---|
| **Hyperopt** | 14343.19 | 119.76 | 4340.16 | 65.88 |

Analysis: Comparable performance to Experiment 4, with a slight increase in test errors but maintaining a good balance between train and test performance.

Conclusion: The final model achieved through Hyperopt provides a promising balance of accuracy and generalisation, aligning well with the business objective of predicting airfare flights accurately.



## b. Business Impact

The release of the final model optimised through serial experimentation and gradual tuning, stands to have a considerable positive impact on the business. By achieving a lower error rate in predicting airfare flights, the model enhances the accuracy and reliability of fare predictions, which is desirable for overall customer satisfaction and trust.

This improvement can lead to increased usage of the service, as customers are more likely to return and recommend a platform that provides dependable predictions. Furthermore, accurate fare prediction aids in better inventory management and pricing strategies for the business, potentially increasing revenue and market competitiveness. The precision of this model can serve as a benchmark for future predictive modelling endeavours within the company.

## c. Encountered Issues

### Resolving time limitations

As it stands, the XGBoost model has taken the longest to carry out, compared to the KNN and Linear models.

No additional errors to report.

## FUTURE EXPERIMENT

### a. Key Learning

*< Data Insights from EDA remain unchanged from Experiment 1. >*

### b. Suggestions / Recommendations

For a feature-rich and large dataset, LightGBM offers an excellent balance of speed, efficiency, and accuracy, one that may be suitable for exploration next. The potential risks of overfitting and the complexity of model tuning and interpretation are important considerations. Its suitability would depend on the specific requirements of the dataset and the capacity to manage and fine-tune the model effectively.

---

#### Points For LightGBM

Efficiency with Large Datasets:
LightGBM is designed to be efficient with large data sets. It uses histogram-based algorithms, which bucket continuous feature values into discrete bins, significantly speeding up training and reducing memory usage.

Handling of High Dimensionality:
It is well-suited for datasets with a large number of features. LightGBM can handle the complexity and intricacies of extensive feature spaces without a significant decrease in training speed.

Optimised Performance:
LightGBM uses Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB), which contributes to its faster training times and lower memory consumption, while still maintaining high model accuracy.

Flexible and Robust:
It offers robust handling of overfitting and gives better results on unbalanced data sets. Its flexibility allows for fine-tuning of hyperparameters and supports various loss functions.

#### Points Against LightGBM

Overfitting Risk with Small Datasets:
While excellent with large datasets, LightGBM can overfit when used with smaller datasets. It requires careful tuning of parameters to avoid overfitting.

Complexity in Interpretation and Tuning:
The complexity of the model makes it harder to interpret compared to simpler models like linear regression. Also, hyperparameter tuning can be more complex and require a deeper understanding of the model's workings.

Sensitivity to Overfitting with Noisy Data:
LightGBM can be sensitive to overfitting, especially in the presence of noisy data, which requires careful regularization and parameter tuning.

#### Deployment

While the model has the capacity to make relatively accurate predictions, deployment is not advised until an investigation into other modelling frameworks has been conducted. The rationale for this is to maximise the business' return on investment, whereas other models may provide a different method of exploring and interpreting the dataset's features.